

Scalable and Robust Construction of Topical Hierarchies

Chi Wang, Xueqing Liu, Yanglei Song, Jiawei Han
 Department of Computer Science
 University of Illinois at Urbana-Champaign
 Urbana, IL, USA
 {chiwang1, xliu93, ysong44, hanj}@illinois.edu

ABSTRACT

Automated generation of high-quality topical hierarchies for a text collection is a dream problem in knowledge engineering with many valuable applications. In this paper a scalable and robust algorithm is proposed for constructing a hierarchy of topics from a text collection. We divide and conquer the problem using a top-down recursive framework, based on a tensor orthogonal decomposition technique. We solve a critical challenge to perform scalable inference for our newly designed hierarchical topic model. Experiments with various real-world datasets illustrate its ability to generate robust, high-quality hierarchies efficiently. Our method reduces the time of construction by several orders of magnitude, and its robust feature renders it possible for users to interactively revise the hierarchy.

1. INTRODUCTION

Automated, hierarchical organization of the concepts in a textual database at different levels of granularity is an important problem in knowledge engineering with many valuable applications such as information summarization, search and online analytical processing (OLAP). With vast amount of text data and dynamic change of users' need, it is too costly to rely on human experts to do manual annotation and provide ready-to-use topical hierarchies. Thus it is critical to create a robust framework for automated construction of high-quality topical hierarchies from texts in different domains.

Limitation of prior work. A main body of existing work uses a bag-of-words topic modeling approach to model word occurrences in the documents. They infer the whole hierarchy all at once by inference methods such as Gibbs sampling. Such methods have the following bottlenecks:

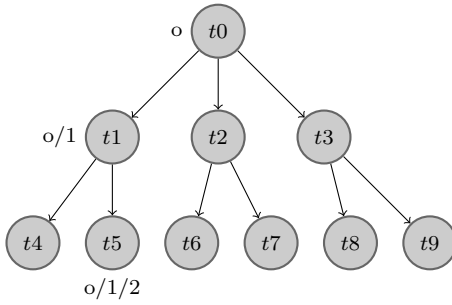
1. **Scalability.** The inference methods such as Gibbs sampling are expensive, requiring multiple passes of the data. The number of passes has no theoretical bound, and typically needs to be several hundreds or thousands.

2. **Robustness.** The inference algorithms do not produce a unique solution in nature. The variance of different algorithm runs can be very large especially when the hierarchy is deep. This prevents a user from revising the local structure of a hierarchy (e.g., changing the number of branches of one node).
3. **Interpretability.** The unigram representation of topics has limited human interpretability, especially for very specific topics. But it is challenging to integrate multigrams into topic hierarchy modeling in a scalable and robust manner.

Insight. The following ideas lead to our proposed solution.

1. *Consider a strategy of top-down recursive construction of a topical hierarchy, instead of inferring a complex hierarchical model all at once.* For example, in Figure 1, we first infer the topics t_1 to t_3 at the first level and then infer subtopics for each of them. Thus the problem of uncovering a big tree can be divided into a set of subproblems: uncovering subtrees, and then conquered by solving each subproblem in the same manner. Then we can focus on solving a simpler (yet still challenging) problem: uncovering topics for one level, with good scalability, robustness and interpretability.
2. *Compress the original data by collecting important statistics from the documents, e.g., word co-occurrences, in order to infer the topics efficiently and robustly.* For one advantage, the inference based on the compressed information avoids the expensive, numerous passes of the data. For another advantage, the compression reduces certain degree of randomness in the data. By carefully choosing the statistics and the inference method, we can uncover the topics with theoretical guarantee. This insight is supported by some very recent finding in the theory study [4], and we leverage it as a most important basis to develop our recursive approach and justify it.
3. *Enhance the topic representation using frequent multigrams, mined from the documents and placed in the topic hierarchy according to the inferred topics.* We do not consider multigrams when inferring topics, due to the scalability consideration. But we perform lightweight posterior estimation of the distribution over the topics for each frequent multigram. Then we can sort the multigrams as topical phrases for each topic. For example, a database topic should contain 'database system,' 'query processing' etc. as top-ranked phrases.

We systematically develop our solution based on these insights. The following summarizes our main contributions:



Topic	Word distribution	Representative phrase
t0(o)	data:0.01, learning:0.01 ...	database system, machine learning ...
t1(o/1)	database:0.05, system:0.01 ...	database system, management ...
t2(o/2)	information:0.1, retrieval:0.05 ...	information retrieval, web search ...
t3(o/3)	learning:0.11, classification:0.01 ...	learning, classification, feature selection ...
t4(o/1/1)	query:0.12, processing:0.07 ...	query processing, query optimization ...
t5(o/1/2)	system:0.08, distributed:0.03 ...	distributed database, concurrency control ...

Figure 1: An example of the topical hierarchy. Each topic can be denoted by the path from root topic to it

- We propose a new hierarchical topic model, which supports divide-and-conquer inference as mentioned above. We provide theoretical justification of doing so.
- We develop a scalable tensor-based recursive orthogonal decomposition (STROD) method to infer the model. It inherits the nice theoretical properties of the tensor orthogonal decomposition algorithm [4], but has significantly better scalability. To the best of our knowledge, it is the first scalable and robust algorithm for topical hierarchy construction.
- Our experiments demonstrate that our method can scale up to datasets that are orders of magnitude larger than the state-of-the-art, while generating quality topic hierarchy that is comprehensible to users.

2. PROBLEM FORMULATION

The input is a corpus of D documents. Every document d_i can be segmented into a sequence of l_i tokens: $d_{i,j}, j = 1, \dots, l_i$. For convenience we index all the unique words in this corpus using a vocabulary of V words. And $d_{i,j} = x, x \in \{1, \dots, V\}$ means that the j -th token in document d_i is the x -th word in the vocabulary. Throughout this paper we use ‘word x ’ to refer to the x -th word in the vocabulary.

Given a corpus, our goal is to construct a topical hierarchy. A topical hierarchy is a tree of topics, where each child topic is about a more specific theme within the parent topic. Statistically, a topic t is characterized by a probability distribution over words ϕ_t . $\phi_{t,x} = p(x|t) \in [0, 1]$ is the probability of seeing the word x in topic t , and $\sum_{x=1}^V \phi_{t,x} = 1$. For example, in a topic about the database research area, the probability of seeing “database”, “system” and “query” is high, and the probability of seeing “speech”, “handwriting” and “animation” is low. This characterization is advantageous in statistical modeling of text, but is weak in human interpretability, due to two reasons. First, unigrams may be ambiguous, especially across specific topics. Second, the probability $p(x|t)$ reflects the popularity of a word x in the topic t , but not its discriminating power. For example, a general word “algorithm” may have higher probability than a discriminative word “locking” in the database topic.

For these reasons, we choose to enhance the topic representation with ranked phrases. Phrases reduce the ambiguity of unigrams. And the ranking should reflect both their popularity and discriminating power for a topic. For example, the database topic can be described as: {“database systems”, “query processing”, “concurrency control”, ...}.

A phrase can appear in multiple topics, though it will have various ranks in them.

Formally, we define a topical hierarchy as follows.

DEFINITION 1 (TOPICAL HIERARCHY). *A topical hierarchy is defined as a tree \mathcal{T} in which each node is a topic. Every non-leaf topic t has C_t child topics. Topic t is characterized by a probability distribution over words ϕ_t , and presented as an ordered list of phrases $\mathcal{P}_t = \{P_{t,1}, P_{t,2}, \dots\}$, where $P_{t,i}$ is the phrase ranked at i -th position for topic t .*

The number of child topics C_t of each topic can be specified as input, or decided automatically by the construction method. In both cases, we assume it is bounded by a small number K , such as 10. This is for efficient browsing of the topics. The number K is named the *width* of the tree \mathcal{T} .

For convenience, we denote a topic using the top-down path from root to this topic. The root topic is denoted as o . Every non-root topic t is denoted by π_t/χ_t , where π_t is the notation of its parent topic, and χ_t is the index of t among its siblings. For example, the topic $t1$ in Figure 1 is denoted as $o/1$, and its child $t5$ is denoted as $o/1/2$. The *level* h_t of a topic t is defined to be the number of ‘/’ in its notation. So root topic is in level 0, and $t5$ is in level 2. The *height* H of a tree is defined to be the maximal level over all the topics in the tree. Clearly, the total number T of topics is upper bounded by $\frac{K^{H+1}-1}{K-1}$.

2.1 Desired Property

The following are the desired properties for a topical hierarchy construction method.

1. Scalability. The scale of the problem is determined by these variables: the number of documents D , the vocabulary size V , the total length of documents L , the total number of topics T , and the width of the topical hierarchy K . These variables are not independent. For example, the average length of documents L/D should be larger than 1, and the number of documents D is usually much larger than the vocabulary size V . Typically, the number of tokens L is the dominant factor. For scalability the algorithm must have sublinear complexity with respect to L . When the dataset is too large to fit in memory, an ideal algorithm should only perform a small constant number of passes of the data.

2. Robustness. A construction algorithm is robust in the following senses.

PROPERTY 1 (ROBUST RECOVERY). *Suppose the data is generated from certain topic word distributions exactly following a generative process, the recovery is robust if the exact distributions can be found when sufficient data are given.*

For example, in Figure 1, if sufficient data are generated from the topic word distributions as shown on the right hand side, an robust recovery algorithm should return these distributions rather than other distributions.

In certain scenarios, part of the constructed hierarchy needs to be revised to customize for users’ need. For example, one may want to change the number of branches or height of a subtree.

PROPERTY 2 (ROBUST REVISION). *The revision to a subtree $\mathcal{T}(t)$ rooted at topic t is robust, if every topic t' not in the subtree $\mathcal{T}(t)$ remains intact word distribution in the returned hierarchy.*

In Figure 1, if one wants to partition topic $t1$ into 3 subtopics instead of 2, but also wants to keep other parts of the tree intact, a robust algorithm should not change the output of topic $t2, t3$ and $t6$ to $t10$. This property assures that the local change to a large hierarchy doesn’t alter the remainder of the tree.

3. Interpretability. There are two aspects for the interpretability of a topical hierarchy. i) Topic coherence: one can interpret the meaning of an individual topic given the ranked words and phrases from it; and ii) Parent-child relationship: one can interpret the meaning of the edges between a parent topic and its child topics given the ranked words and phrases from them.

3. RELATED WORK

Statistical topic modeling techniques model documents as mixtures of multiple topics, while every topic is modeled as a distribution over words. Two important models are PLSA (probabilistic latent semantic analysis) [15] and its Bayesian extension LDA (latent Dirichlet allocation) [6]. They model the generative processes of the words for all the documents in a corpus. To generate each word, a latent topic label is first chosen from a pool of ‘flat topics’ with a multinomial distribution. And then a word is sampled according to this topic’s word distribution. With these generative assumptions, the unknown word distribution of every topic can be inferred so as to best explain the observed word occurrences in the documents.

Hierarchical topic models follow the same generative spirit. Instead of generating from a pool of flat topics, these models assume an internal hierarchical structure of the topics. Different models use different generative processes to simulate this hierarchical structure: nested Chinese Restaurant Process [11], Pachinko Allocation [17], Hierarchical Pachinko Allocation [21], recursive Chinese Restaurant Process [16], and nested Chinese Restaurant Franchise [2]. When these models are applied to constructing a topical hierarchy, the entire hierarchy must be inferred all at once from the corpus. They do not have the robust revision property.

The main inference methods for these topic models can be divided into two categories: Gibbs sampling [12] and variational inference [6]. They are essentially based on the Maximum Likelihood (ML) principle (in the general sense): find the best parameters that maximize the joint probability specified by a model. There has been a substantial amount

of work on speeding up LDA inference, e.g., by leveraging sparsity [23, 31, 14] and parallelization [22, 25, 32], or online learning mechanism [1, 13, 9]. Few of these ideas have been adopted by the hierarchical topic model studies.

These inference methods have no theoretical guarantee of convergence within a bounded number of iterations, and are nondeterministic either due to the sampling or the random initialization. Recently, a new inference method for LDA has been proposed based on a *method of moments*, rather than ML. It is found to have nice convergence properties in theory [4]. However, the practical issue of scalability has not been solved in the theoretical work.

All of the hierarchical topic models follow the bag-of-words assumption, while some other extensions of LDA have been developed to model sequential n-grams [27, 30, 18]. No one has integrated them in a hierarchical topic model. It is obvious that the scalability and robustness issues will become more challenging in an integrated model. A practical approach is to separate the topic modeling part and the phrase mining part. Blei and Lafferty [5] have proposed to use a statistical test to find topical phrases, which is time-consuming. A much less expensive heuristic is studied in our previous work [8] and shown to be effective.

Finally, we briefly review a few alternative approaches to constructing a topical hierarchy. Pujara and Skomoroch [24] proposed to first run LDA on the entire corpus, and then split the corpus heuristically according to the results and run LDA on each split corpus individually. It is a recursive approach without an integrated generative process, so the robust recovery property is not applicable. Recursive clustering is used to cluster documents [10], queries [19], keywords [29] etc., to construct hierarchies of different kinds. CATHY [29] is a recursive topical phrase mining framework, where the phrase mining and the topic discovery are also separated for efficiency purpose. It uses a word co-occurrence network to compress the documents and performs topic discovery using an EM algorithm. However, it is designed for short, content-representative text and also suffers the scalability and robustness issues.

4. HIERARCHICAL TOPIC MODELING

In this section, we model how the documents are generated when the topical hierarchy is given. Based on that we develop our solution in the next section. Our hierarchical topic model is a unigram-based model that supports recursive inference. We first explain the motivation behind the model in Section 4.1, and then present it in Section 4.2.

4.1 Motivation

First, we advocate using a top-down recursive framework to construct the hierarchy level by level, instead of all at once. Compared with a gigantic non-recursive construction method, a recursive framework has the following advantages: (i) it facilitates robust revision; (ii) the scalability and robust recovery challenges of the topical hierarchy can be reduced by the divide-and-conquer philosophy; and (iii) parent-child relationship naturally follows the recursion. However, it may have the following disadvantages: (i) it misses some complex dependency across the topics mentioned within a document; and (ii) the error in recovering a parent topic may propagate to a child topic, which implies the critical importance of robust recovery for top levels. Since our major concern is to discover the recurring topics of the corpus rather than each

Table 1: Notations used in our model

Symbol	Description
D	the number of documents in the corpus
V	the number of unique words in the corpus
$d_{i,j}$	the j -th word in the i -th document
l_i	the length (number of tokens) of document d_i
L	the total number of tokens in the corpus $\sum_{i=1}^D l_i$
π_t	the parent topic of topic t
χ_t	the suffix of topic t 's notation ($t = \pi_t/\chi_t$)
C_t	the number of child topics of topic t
o	the root topic
ϕ_t	the multinomial distribution over words in topic t
α_t	the Dirichlet hyperparameter vector of topic t
$\theta_{i,t}$	the distribution over child topics of t for d_i
T	the total number of topics in the hierarchy
τ	the number of leaf topics in the hierarchy

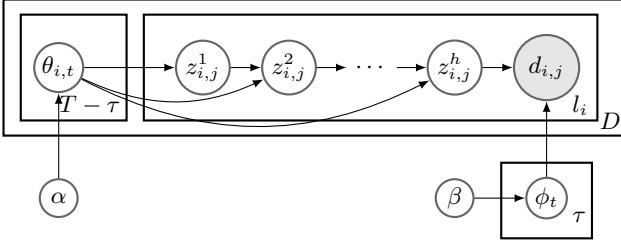


Figure 2: Latent Dirichlet Allocation with Topic Tree

single document, we are willing to use a simple tree structure for better scalability. Meanwhile, we need to carefully design our method to have good robustness and interpretability.

Second, for scalability consideration, we do not model the n -grams in the model, but discover topical phrases separately after the model inference. This strategy is shown to be effective in previous work [5, 28]. We discuss the phrase mining and ranking in Section 5.3.

Based on these considerations, we propose a new hierarchical topic model that allows for scalable, robust recursive inference.

4.2 Latent Dirichlet Allocation with Topic Tree

In our model, every document is modeled as a series of multinomial distributions: one multinomial distribution for every non-leaf topic over its child topics, representing the content bias towards the subtopics. For example, in Figure 1, there are 4 non-leaf topics: $o, o/1, o/2$ and $o/3$. So a document d_i is associated with 4 multinomial topic distributions: $\theta_{i,o}$ over its 3 children, and $\theta_{i,o/1}, \theta_{i,o/2}, \theta_{i,o/3}$ over their 2 children each. When the height of the hierarchy $H = 1$, it reduces to the flat LDA model, because only the root is a non-leaf node. Each multinomial distribution $\theta_{i,t}$ is generated from a Dirichlet prior α_t . $\alpha_{t,z}$ represents the corpus bias towards z -th child of topic t , and $\alpha_{t,0} = \sum_{z=1}^{C_t} \alpha_{t,z}$.

For every leaf topic node t , $C_t = 0$, and a multinomial distribution ϕ_t over words is generated from another Dirichlet prior β . These word distributions are shared by the entire corpus.

To generate a word $d_{i,j}$, we first sample a path from the root to a leaf node $o/z_{i,j}^1/z_{i,j}^2/\dots/z_{i,j}^h$. The nodes along the path are sampled one by one, starting from the root. Each time one child $z_{i,j}^k$ is selected from all children of

$o/z_{i,j}^1/\dots/z_{i,j}^{k-1}$, according to the multinomial $\theta_{i,o/z_{i,j}^1/\dots/z_{i,j}^{k-1}}$.

When a leaf node is reached, the word is generated from the multinomial distribution $\phi_{o/z_{i,j}^1/z_{i,j}^2/\dots/z_{i,j}^h}$. Note that the length of the path h is not necessary to be equal for all documents, if not all leaf nodes are on the same level.

The whole generative process is illustrated in Figure 2. Table 1 collects the notations.

A feature supporting recursive construction. For every non-leaf topic node, we can derive a word distribution by marginalizing their child topic word distributions:

$$\phi_{t,x} = p(x|t) = \sum_{z=1}^{C_t} p(z|t)p(x|t/z) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z,x} \quad (1)$$

So in our model, the word distribution ϕ_t for an internal node in the topic hierarchy can be seen as a mixture of its child topic word distributions. The Dirichlet prior α_t determines the mixing weight.

A topical hierarchy \mathcal{T} is parameterized by $\alpha_t(\mathcal{T})$ where $C_t(\mathcal{T}) > 0$, and $\phi_t(\mathcal{T})$ where $C_t(\mathcal{T}) = 0$. We define a topical hierarchy \mathcal{T}_1 to be *subsumed* by \mathcal{T}_2 , if there is a mapping κ from node t in \mathcal{T}_1 to node t' in \mathcal{T}_2 , such that for every node t in \mathcal{T}_1 , $\pi_t(\mathcal{T}_1) = \pi_{\kappa(t)}(\mathcal{T}_2)$, and one of the following is true:

1. $C_t(\mathcal{T}_1) = C_{\kappa(t)}(\mathcal{T}_2) > 0$ and $\alpha_t(\mathcal{T}_1) = \alpha_{\kappa(t)}(\mathcal{T}_2)$; or
2. $C_t(\mathcal{T}_1) = 0$ and $\phi_t(\mathcal{T}_1) = \phi_{\kappa(t)}(\mathcal{T}_2)$.

In words, a subsumed tree is obtained by removing all the descendants of some nodes in a larger tree, and absorbing the word distributions of the descendants into the new leaf nodes. In Figure 3, we show three trees and each tree is subsumed by the one on its right. The subsumed tree retains equivalent high-level topic information of a larger tree, and can be recovered before we recover the larger tree. This idea allows us to recursively construct the whole hierarchy, which distinguishes our method from all-at-once construction methods. We present the recursive construction approach with justification in the next section.

5. THE STROD FRAMEWORK

We propose a Scalable Tensor Recursive Orthogonal Decomposition (STROD) framework for topical hierarchy construction, the first that meets all the criteria in Section 2.1. It uses tensor (hypermatrix) decomposition to perform moment-based inference of the hierarchical topic model proposed in Section 4 recursively.

5.1 Moment-based Inference

The central idea of the inference method is based on the *method of moments*, instead of *maximum likelihood*. It enables tractable computations to estimate the parameters.

In statistics, the *population moments* are expected values of powers of the random variable under consideration. The method of moments derives equations that relate the population moments to the model parameters. Then, it collects empirical population moments from observed samples, and solve the equations using the sample moments in place of the theoretical population moments. In our case, the random variable is the word occurring in each document. The population moments are expected occurrences and co-occurrences of the words. They are related to the model parameters α and ϕ . We can collect empirical population moments from

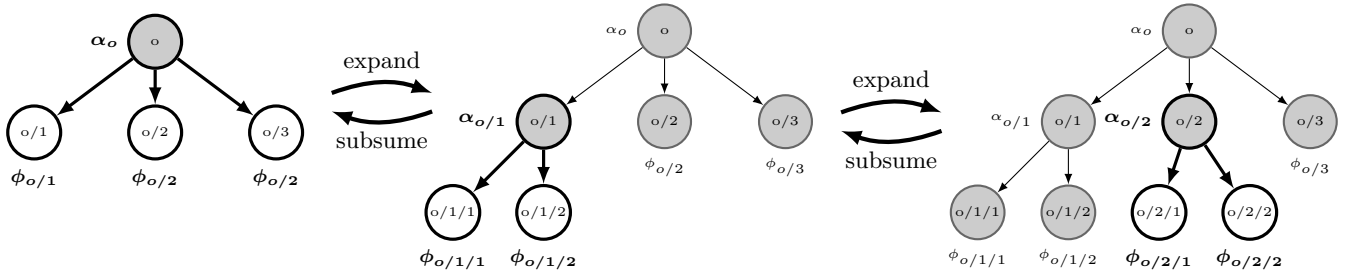


Figure 3: An illustration of recursive topical hierarchy construction. The construction order is from left to right. Each time one leaf topic node is expanded into several child topics (unshaded) and the relevant parameters (in bold) are estimated. The same figure explains *subsumption* relationship: A tree on the left is subsumed by a tree on the right

the corpus, and estimate α and ϕ by fitting the empirical moments with theoretical moments. One particular computational advantage is that the inference only relies on the empirical population moments (word co-occurrence statistics). They compress important information from the full data, and require only one scan of the data to collect.

The idea is promising, but not straightforward to apply to our model. The challenge is that the same population moments can be expressed by parameters on different levels. For the example in Figure 1, we can derive equations of the population moments (expected word co-occurrences) based on the model parameters associated with $t1, t2$ and $t3$, or based on those with $t4 - t9$. Solving these equations independently will find 3 general topics and 6 more specific topics, but will neither reveal their relationship, nor guarantee the existence of the relationship.

Below we present a recursive inference method step by step and justify its correctness.

1. Conditional population moments. We consider the population moments *conditioned* on a non-leaf topic t . The first order moment is the expectation of word x 's occurrence given that it is drawn from topic t 's descendant. We have $p(x|t, \alpha) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z,x}$ according to Equation (1).

The second order moment is the expectation of the co-occurrences of two words x_1 and x_2 given that they are both drawn from topic t 's descendants.

$$p(x_1, x_2 | t, \alpha) = \sum_{z_1 \neq z_2} \frac{\alpha_{t,z_1} \alpha_{t,z_2}}{\alpha_{t,0}(\alpha_{t,0} + 1)} \phi_{t/z_1,x_1} \phi_{t/z_2,x_2} + \sum_{z=1}^{C_t} \frac{\alpha_{t,z}(\alpha_{t,z} + 1)}{\alpha_{t,0}(\alpha_{t,0} + 1)} \phi_{t/z,x_1} \phi_{t/z,x_2} \quad (2)$$

Likewise, we can derive the third order moment as the expectation of co-occurrences of three words x_1, x_2 and x_3 given that they are all drawn from topic t 's descendants:

$$\begin{aligned} p(x_1, x_2, x_3 | t, \alpha) &= \sum_{z_1 \neq z_2 \neq z_3} \frac{\alpha_{t,z_1} \alpha_{t,z_2} \alpha_{t,z_3}}{\alpha_{t,0}(\alpha_{t,0} + 1)(\alpha_{t,0} + 2)} \phi_{t/z_1,x_1} \phi_{t/z_2,x_2} \phi_{t/z_3,x_3} \\ &+ \sum_{z_1 \neq z_2} \frac{\alpha_{t,z_1} \alpha_{t,z_2}(\alpha_{t,z_1} + 1)}{\alpha_{t,0}(\alpha_{t,0} + 1)(\alpha_{t,0} + 2)} (\phi_{t/z_1,x_1} \phi_{t/z_1,x_2} \phi_{t/z_2,x_3} \\ &+ \phi_{t/z_1,x_1} \phi_{t/z_1,x_3} \phi_{t/z_2,x_2} + \phi_{t/z_1,x_3} \phi_{t/z_2,x_2} \phi_{t/z_2,x_1}) \\ &+ \sum_{z=1}^{C_t} \frac{\alpha_{t,z}(\alpha_{t,z} + 1)(\alpha_{t,z} + 2)}{\alpha_{t,0}(\alpha_{t,0} + 1)(\alpha_{t,0} + 2)} \phi_{t/z,x_1} \phi_{t/z,x_2} \phi_{t/z,x_3} \end{aligned} \quad (3)$$

These equations exhibit good opportunities for a recursive solution, because the moments conditioned on a topic t can be expressed by only the Dirichlet prior and word distributions associated with its child topics. If these low order moments can uniquely determine the model parameters, we can use them to recover the child topics of every topic robustly, and by recursion, we can then construct the whole tree (Figure 3).

Fortunately, there is indeed a robust technique to recover the parameters from low order moments.

2. Tensor orthogonal decomposition. Anandkumar *et al.* [4] proved that with some mild non-degeneracy conditions, the following equations can be uniquely solved by a tensor orthogonal decomposition method:

$$M_2 = \sum_{z=1}^k \lambda_z v_z \otimes v_z, M_3 = \sum_{z=1}^k \lambda_z v_z \otimes v_z \otimes v_z \quad (4)$$

where M_2 is a $V \times V$ tensor (hence, a matrix) and M_3 is a $V \times V \times V$ tensor, λ_z is an unknown positive value about the weight of z -th component v_z , which is an unknown V -dimensional vector. In words, both M_2 and M_3 can be decomposed into the same number of components, and each component is determined by a single vector. The operator \otimes denotes an outer product between tensors: if $A \in \mathbb{R}^{s_1 \times \dots \times s_p}$, and $B \in \mathbb{R}^{s_{p+1} \times \dots \times s_{p+q}}$, then $A \otimes B$ is a tensor in $\mathbb{R}^{s_1 \times \dots \times s_{p+q}}$, and $[A \otimes B]_{i_1 \dots i_{p+q}} = A_{i_1 \dots i_p} B_{i_{p+1} \dots i_{p+q}}$.

To write Equations (1)-(3) in this form, we define:

$$M_1(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z} \quad (5)$$

$$E_2(t) = [p(x_1, x_2 | t, \alpha)]_{V \times V} \quad (6)$$

$$M_2(t) = (\alpha_{t,0} + 1)E_2(t) - \alpha_{t,0}M_1(t) \otimes M_1(t) \quad (7)$$

$$E_3(t) = [p(x_1, x_2, x_3 | t, \alpha)]_{V \times V \times V} \quad (8)$$

$$U_1(t) = E_2(t) \otimes M_1(t), \quad (9)$$

$$U_2(t) = \Omega(U_1(t), 1, 3, 2), U_3(t) = \Omega(U_1(t), 2, 3, 1)$$

$$M_3(t) = \frac{(\alpha_{t,0} + 1)(\alpha_{t,0} + 2)}{2} E_3(t) + \alpha_{t,0}^2 M_1 \otimes M_1 \otimes M_1 - \frac{\alpha_{t,0}(\alpha_{t,0} + 1)}{2} [U_1(t) + U_2(t) + U_3(t)] \quad (10)$$

where $\Omega(A, a, b, c)$ permutes the modes of tensor A , such that $\Omega(A, a, b, c)_{i_1, i_2, i_3} = A_{i_a, i_b, i_c}$. It follows that:

$$M_2(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z} \otimes \phi_{t/z}, M_3(t) = \sum_{z=1}^{C_t} \frac{\alpha_{t,z}}{\alpha_{t,0}} \phi_{t/z} \otimes \phi_{t/z} \otimes \phi_{t/z}$$

So they fit Equation (4) nicely, and intuitively. If we decompose $M_2(t)$ and $M_3(t)$, the z -th component is determined by

the child topic word distribution $\phi_{t/z}$, and its weight is $\frac{\alpha_{t,z}}{\alpha_{t,0}}$, which is equal to $p(t/z|\alpha_t)$.

Algorithm 1: Tensor Orthogonal Decomposition (TOD)

Input: Tensor $M_2 \in \mathbb{R}^{V \times V}$, $M_3 \in \mathbb{R}^{V \times V \times V}$, number of components k , number of outer and inner iterations N and n

Output: The decomposed components $(\lambda_z, v_z), z = 1, \dots, k$

```

1.1 Compute  $k$  orthonormal eigenpairs  $(\sigma_z, \mu_z)$  of  $M_2$ ;
1.2 Compute a whitening matrix  $W = M\Sigma^{-\frac{1}{2}}$ ;
    //  $M = [\mu_1, \dots, \mu_k], \Sigma = \text{diag}(\sigma_1, \dots, \sigma_k), W^T M_2 W = I$ 
1.3 Compute  $(W^T)^+ = M\Sigma^{\frac{1}{2}}$ ;
    // the Moore-Penrose pseudoinverse of  $W^T$ 
1.4 Compute a  $k \times k \times k$  tensor  $\tilde{T} = M_3(W, W, W)$ ;
    //  $\tilde{T}_{i1,j1,k1} = \sum_{i2,j2,k2} (M_3)_{i2,j2,k2} W_{i2,i1} W_{j2,j1} W_{k2,k1}$ 
1.5 for  $z = 1..k$  do
1.6    $\lambda^* \leftarrow 0$ ; // the largest eigenvalue so far
1.7   for  $\text{outerIter} = 1..N$  do
1.8      $v \leftarrow$  a random unit-form vector;
1.9     for  $\text{innerIter} = 1..n$  do  $v \leftarrow \frac{\tilde{T}(I, v, v)}{\|\tilde{T}(I, v, v)\|}$  // power
        iteration update
1.10    if  $\tilde{T}(v, v, v) > \lambda^*$  then  $(\lambda^*, v^*) \leftarrow (\tilde{T}(v, v, v), v)$ 
        // choose the largest eigenvalue
1.11    end
1.12     $\lambda_z = \frac{1}{(\lambda^*)^2}, v_z = \lambda_z (W^T)^+ v^*$ ;
    // recover eigenpair of the original tensor
1.13     $\tilde{T} \leftarrow \tilde{T} - \lambda^* v^* \otimes v^* \otimes v^*$ ; // deflation
1.14 end
1.15 return  $(\lambda_z, v_z), z = 1, \dots, k$ 

```

Algorithm 1 outlines the tensor orthogonal decomposition method for recovering the components. It can be partitioned into two parts:

- Lines 1.1 to 1.4 project the large tensor $M_3 \in \mathbb{R}^{V \times V \times V}$ into a smaller tensor $\tilde{T} \in \mathbb{R}^{k \times k \times k}$. \tilde{T} is not only of smaller size, but can be decomposed into an orthogonal form: $\tilde{T} = \sum_{z=1}^k \tilde{\lambda}_z \tilde{v}_z^{\otimes 3}$. $\tilde{v}_z, i = 1, \dots, k$ are orthonormal vectors in \mathbb{R}^k . This is assured by the whitening matrix W calculated in Line 1.2, which satisfies $W^T M_2 W = I$.
- Lines 1.5 to 1.14 perform orthogonal decomposition of \tilde{T} via a power iteration method. The orthonormal eigenpairs $(\tilde{\lambda}_z, \tilde{v}_z)$ are found one by one. To find one such pair, the algorithm randomly starts with a unit-form vector v , runs power iteration (Line 1.9) for n times, and records the candidate eigenpair. This process further repeats by N times, starting from different unit-form vectors, and the candidate eigenpair with the largest eigenvalue is picked (Line 1.10). After an eigenpair is found, the tensor \tilde{T} is deflated by the found component (Line 1.13), and the same power iteration is applied to it to find the next eigenpair. After all the k orthonormal eigenpairs $(\tilde{\lambda}_z, \tilde{v}_z)$ are found, they can be used to uniquely determine the k target components (λ_z, v_z) (Line 1.12).

The following theorem ensures that the decomposition is unique and fast.

THEOREM 1. Assume M_2 and M_3 are defined as in Equation 4, $\lambda_z > 0$, and the vectors v_z 's are linearly independent

and have unit-form, then Algorithm 1 returns exactly the same set of (λ_z, v_z) . Furthermore, the power iteration step of Line 1.9 converges in a quadratic rate.

This theorem is essentially a combination of Theorem 4.3 and Lemma 5.1 in Anandkumar *et al.* [4]. See Appendix A for more discussion about it. The importance of Theorem 1 is that it allows us to use moments only up to the third order to recover the exact components, and the convergence is fast.

3. Recursive decomposition. With Algorithm 1 as a building block, we can divide and conquer the inference of the whole model. We devise Algorithm 2, which recursively infers model parameters in a top-down manner. Taking any topic node t as input, it computes the conditional moments $M_2(t)$ and $M_3(t)$. If t is not root, they are computed from the parent topic's moments and estimated model parameters. For example, according to Bayes's theorem,

$$\begin{aligned}
[E_2(t)]_{x_1, x_2} &= p(x_1, x_2 | t, \alpha) \propto p(x_1, x_2, t | \pi_t, \alpha) \\
&= p(x_1, x_2 | \pi_t, \alpha) p(t | x_1, x_2, \pi_t, \alpha) \\
&= [E_2(\pi_t)]_{x_1, x_2} \alpha_{\pi_t, \chi_t} (\alpha_{\chi_t, z} + 1) \phi_{t, x_1} \phi_{t, x_2} \\
&\quad / \left(\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t, z} (\alpha_{\pi_t, z} + 1) \phi_{\pi_t/z, x_1} \phi_{\pi_t/z, x_2} \right. \\
&\quad \left. + \sum_{z_1 \neq z_2} \alpha_{\pi_t, z_1} \alpha_{\pi_t, z_2} \phi_{\pi_t/z_1, x_1} \phi_{\pi_t/z_2, x_2} \right)
\end{aligned} \tag{11}$$

Other quantities in Equations (5)-(10) can be computed similarly. Then it performs tensor decomposition and recovers the parameter α_t and $\phi_{t/z}$ for each child topic. It then enumerates its children and makes recursive calls with each of them as input. The recursion stops when reaching leaf nodes, where $C_t = 0$. A call of Algorithm 2 with the root o as input will construct the entire hierarchy.

Algorithm 2: Recursive Tensor Orthogonal Decomposition (RTOD)

Input: topic t , number of outer and inner iterations N, n

```

2.1 Compute  $M_2(t)$  and  $M_3(t)$ ;
    // only relies on  $t$ 's ancestors
2.2  $(\lambda_z, v_z) \leftarrow \text{TOD}(M_2(t), M_3(t), C_t, N, n)$ ;
2.3  $\alpha_{t,z} = \alpha_{t,0} \lambda_z, \phi_{t/z} = v_z$ ;
2.4 for  $z = 1..C_t$  do
2.5   RTOD( $t/z, N, n$ );
    // Recursion, get the parameters for the
    subtree rooted at each child
2.6 end

```

The correctness of this recursive algorithm is permitted by the special structure of our model. In particular, we have the following theorem.

THEOREM 2. The RTOD algorithm (Algorithm 2) has both robust recovery and robust revision property.

The robust recovery property follows Theorem 1, plus the fact that the conditional moments of a topic can be expressed by only the Dirichlet prior and word distributions associated with its child topics. The robust revision property is due to conditional independence during the recursive construction procedure: i) once a topic t has been visited

in the algorithm, the construction of its children is independent of each other; and ii) the conditional moments $M_2(t)$ and $M_3(t)$ can be computed independently of t 's descendants. In fact, it leads to a stronger claim.

COROLLARY 1. *If \mathcal{T}_1 is subsumed by \mathcal{T}_2 with the mapping $\kappa(\cdot)$, then the RTOD algorithm on \mathcal{T}_1 and \mathcal{T}_2 returns identical parameters for \mathcal{T}_1 and $\kappa(\mathcal{T}_1)$.*

Therefore, the tree topology can be expanded or varied locally with minimal revision to the inferred topics. This is in particular useful when the structure of the topic tree is not fully determined in the beginning. The recursive construction offers users a chance to see the construction results and interact with the topic tree expansion or its local variations by deciding on the number of topics.

5.2 Scalability Improvement

Although Algorithms 1 and 2 are robust, they are not scalable. The orthogonal decomposition of the tensor $\tilde{T} \in \mathbb{R}^{k \times k \times k}$ (Lines 1.5-1.14) is efficient, because k is small. However, the bottleneck of the computation is preparing the tensor \tilde{T} , including Line 2.1 and Lines 1.1 to 1.4. They involve the creation of a dense tensor $M_3 \in \mathbb{R}^{V \times V \times V}$, and the time-consuming operation $M_3(W, W, W)$. Since V is usually tens of thousands or larger, it is impossible to store such a tensor in memory and perform the tensor product operation. In fact, even the second order moment $M_2 \in \mathbb{R}^{V \times V}$ is dense and large, challenging both space and time efficiency already.

Anandkumar [4] discusses a plausible way to solve the space challenge, by avoiding explicit creation of the tensors M_3 and \tilde{T} . It suggests going through the document-word occurrence data for computing the power iterations. This requires as many times of data passes as other inference methods, if not more. Therefore, its scalability will still be unsatisfactory.

We make key contributions to solving the challenge in a different approach. We avoid explicit creation of both tensor M_3 and M_2 . But we do explicitly create \tilde{T} since it is memory efficient. Therefore, the efficient power iteration updates remain as in Algorithm 1. Utilizing the special structure of the tensors in our problem, we show that \tilde{T} can be created by passing the data only twice, without incurring creations of any dense V^2 or V^3 tensors.

1. Avoid creating M_2 . For ease of discussion, we omit the conditional topic t in the notation of this and next subsection. According to Equation (7), $M_2 = (\alpha_0 + 1)E_2 - \alpha_0 M_1 \otimes M_1$. E_2 is a sparse symmetric matrix because only two words co-occurring in one document will contribute to the empirical estimation of E_2 . However, $M_1 \otimes M_1$ is a full V by V matrix. We would like to compute the whitening matrix W without explicit creation of M_2 .

First, we notice that M_1 is in the *column space* of M_2 (i.e., M_1 is a linear combination of M_2 's column vectors), so E_2 has the same column space S as M_2 . Also, since $M_2 = \sum_{z=1}^k \lambda_z v_z \otimes v_z$ is positive definite, so is $E_2 = \frac{1}{\alpha_0 + 1}(M_2 + \alpha_0 M_1 \otimes M_1)$. Let $E_2 = U \Sigma_1 U^T$ be its spectral decomposition, where $U \in \mathbb{R}^{V \times k}$ is the matrix of k eigenvectors, and $\Sigma_1 \in \mathbb{R}^{k \times k}$ is the diagonal eigenvalue matrix. The k column vectors of U form an orthonormal basis of S . M_1 's representation in this basis is $M'_1 = U^T M_1$. Now, M_2 can be written

as:

$$M_2 = U[(\alpha_0 + 1)\Sigma_1 - \alpha_0 M'_1 \otimes M'_1]U^T$$

So, a second spectral decomposition can be performed on $M'_2 = (\alpha_0 + 1)\Sigma_1 - \alpha_0 M'_1 \otimes M'_1$, as $M'_2 = U' \Sigma U'^T$. Then we have:

$$M_2 = U U' \Sigma (U U')^T$$

Therefore, we effectively obtain the spectral decomposition of $M_2 = M \Sigma M^T$ without creating M_2 . Not only the space requirement is reduced (from a dense $V \times V$ matrix to a sparse matrix E_2), but also the time for spectral decomposition. If we perform spectral decomposition for M_2 directly, it requires $O(V^3)$ time complexity. However, using the twice spectral decomposition trick above, we just need to compute the first largest k eigenpairs for a sparse matrix E_2 , and a spectral decomposition for a small matrix $M'_2 \in \mathbb{R}^{k \times k}$. The first decomposition can be done efficiently by a power iteration method or other more advanced algorithms [26]. The time complexity is roughly $O(k \|E_2\|_0)$, where $\|E_2\|_0$ is the number of non-zero elements in E_2 . The second decomposition requires $O(k^3)$ time, which can be regarded as constant since $k \leq K \approx 10$.

2. Avoid creating M_3 . The idea is to directly compute $\tilde{T} = M_3(W, W, W)$ without creating M_3 . This is possible due to the distributive law: $(A + B)(W, W, W) = A(W, W, W) + B(W, W, W)$. The key is to decouple M_3 as a summation of many different tensors, such that the computation of the product between each tensor and W is easy.

We begin with the empirical estimation of E_3 . Suppose we use $c_{i,x}$ to represent the count of word x in document d_i . Then E_3 can be estimated by averaging all the 3-word triples in each document:

$$\begin{aligned} E_3 &= \frac{1}{D} [A_1 - A_2 - \Omega(A_2, 2, 1, 3) - \Omega(A_2, 2, 3, 1) + 2A_3] \\ A_1 &= \sum_{i=1}^D \frac{1}{l_i(l_i-1)(l_i-2)} c_i \otimes c_i \otimes c_i \\ A_2 &= \sum_{i=1}^D \frac{1}{l_i(l_i-1)(l_i-2)} c_i \otimes \text{diag}(c_i) \\ A_3 &= \sum_{i=1}^D \frac{1}{l_i(l_i-1)(l_i-2)} \text{tridiag}(c_i) \end{aligned} \quad (12)$$

where $\text{tridiag}(v)$ is a tensor with vector v on its diagonal: $\text{tridiag}(v)_{i,i,i} = v_i$. Let $s_i = \frac{1}{l_i(l_i-1)(l_i-2)}$. From the fact $(v \otimes v \otimes v)(W, W, W) = (W^T v) \otimes (W^T v) \otimes (W^T v) = (W^T v)^{\otimes 3}$, we can derive:

$$A_1(W, W, W) = \sum_{i=1}^D s_i (W^T c_i)^{\otimes 3} \quad (13)$$

Based on another fact, $(v \otimes M)(W, W, W) = (W^T v) \otimes M(W, W) = (W^T v) \otimes W^T M W$, we can derive:

$$A_2(W, W, W) = \sum_{i=1}^D s_i (W^T c_i) \otimes W^T \text{diag}(c_i) W \quad (14)$$

Let W_x^T be the x -th column of W^T , we have:

$$A_3(W, W, W) = \sum_{x=1}^V \sum_{i=1}^D s_i c_{i,x} (W_x^T)^{\otimes 3} \quad (15)$$

So we do not need to explicitly create E_3 to compute $E_3(W, W, W)$. The time complexity using Equations (13)-(15) is $O(Lk^2)$, which is equivalent to $O(L)$ because k is small.

Using the same trick, we can obtain:

$$U_1(W, W, W) = W^T E_2 W \otimes W^T M_1 \quad (16)$$

$$(M_1 \otimes M_1 \otimes M_1)(W, W, W) = (W^T M_1)^{\otimes 3} \quad (17)$$

Equation (16) requires $O(k^2 \|E_2\|_0)$ time to compute, while $\|E_2\|_0$ can be large. We can further speed it up.

We notice that $W^T M_2 W = I$ by definition. Substituting M_2 with Equation (7), we have:

$$W^T[(\alpha_0 + 1)E_2 - \alpha_0 M_1 \otimes M_1]W = I \quad (18)$$

which is followed by:

$$W^T E_2 W = \frac{1}{(\alpha_0 + 1)} [I + \alpha_0 (W^T M_1)^{\otimes 2}] \quad (19)$$

Plugging Equation (19) into (16) further reduces the complexity of computing $U_1(W, W, W)$ to $O(Vk + k^3)$. $U_2(W, W, W)$ and $U_3(W, W, W)$ can be obtained by permuting $U_1(W, W, W)$'s modes, in $O(k^3)$ time.

Putting these together, we have the following fast computation of $\tilde{T} = M_3(W, W, W)$ by passing the data once:

$$\begin{aligned} \tilde{T} = M_3(W, W, W) &= \frac{(\alpha_{t,0} + 1)(\alpha_{t,0} + 2)}{2} E_3(W, W, W) \\ &- \frac{\alpha_{t,0}(\alpha_{t,0} + 1)}{2} [(U_1 + U_2 + U_3)(W, W, W)] + \alpha_{t,0}^2 (W^T M_1)^{\otimes 3} \end{aligned} \quad (20)$$

which requires $O(Lk^2 + Vk^2 + k^3) = O(L)$ time in total.

3. Estimation of empirical conditional moments. To estimate the empirical conditional moments for topic t , we estimate the ‘topical’ count of word x in document d_i as:

$$c_{i,x}(t) = c_{i,x} p(t|x) = c_{i,x}(\pi_t) \frac{\alpha_{\pi_t, \chi_t} \phi_{t,x}}{\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t, z} \phi_{\pi_t/z, x}} \quad (21)$$

and $c_{i,x}(o) = c_{i,x}$. Then we can estimate M_1 and E_2 using these empirical counts:

$$\begin{aligned} M_1(t) &= \sum_{i=1}^D \frac{1}{l_i(t)} c_i(t) \\ E_2(t) &= \sum_{i=1}^D \frac{1}{l_i(t)(l_i(t) - 1)} [c_i(t) \otimes c_i(t) - \text{diag}(c_i(t))] \end{aligned} \quad (22)$$

where $l_i(t) = \sum_{x=1}^V c_{i,x}(t)$. These enable fast estimation of empirical moments by passing data once.

Finally, we have a scalable tensor recursive orthogonal decomposition algorithm as outlined in Algorithm 3.

We notice that the hyperparameter $\alpha_{t,0}$ and the number of child topics C_t are needed to run the STROD algorithm. We discuss how to learn them automatically in Appendix B.

5.3 Phrase Mining and Ranking

After the word distribution in each topic is inferred, we can then mine and rank topical phrases within each topic. The phrase mining and ranking in STROD largely follow CATHY [28]. Here we briefly present the process.

1. Mining. In this work, a phrase is defined as a consecutive sequence of words. When representing a topic, only frequent phrases are of interest. So we treat each sentence as a

Algorithm 3: Scalable Tensor Recursive Orthogonal Decomposition (STROD)

Input: topic t , number of outer and inner iterations N, n

- 3.1 Compute $M_1(t)$ and $E_2(t)$ according to Equation (22);
 - 3.2 Find k largest orthonormal eigenpairs (σ_z, μ_z) of E_2 ;
 - 3.3 $M'_1 = U M_1(t)$; // $U = [\mu_1, \dots, \mu_k]$, $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_k)$
 - 3.4 Compute spectral decomposition for $M'_2 = (\alpha_{t,0} + 1)\Sigma_1 - \alpha_{t,0} M'_1 \otimes M'_1 = U' \Sigma U'^T$;
 - 3.5 $M = U U'$, $W = M \Sigma^{-\frac{1}{2}}$, $(W^T)^+ = M \Sigma^{\frac{1}{2}}$;
 - 3.6 Compute $\tilde{T} = M_3(W, W, W)$ according to Equation (20);
 - 3.7 Perform power iteration Line 1.5 to 1.14 in Algorithm 1;
 - 3.8 $\alpha_{t,z} = \alpha_{t,0} \lambda_z$, $\phi_{t/z} = v_z$;
 - 3.9 **for** $z = 1..C_t$ **do**
 - 3.10 | STROD($t/z, N, n$);
 - 3.11 **end**
-

‘transaction’, and each word as an ‘item’, and mine frequent consecutive patterns as phrases of arbitrary lengths. To filter out incomplete phrases (e.g., ‘vector machine’ instead of ‘support vector machine’) and frequently co-occurred words that do not make up a meaningful phrase (e.g., ‘learning classification’), CATHY defines two criteria *completeness* and *phraseness* to measure them. Following that principle, we use a statistical test to decide quality phrases, and record the count $c_{i,P}$ of each phrase P in each document d_i .

2. Ranking. After the set of frequent phrases of mixed lengths is mined, they are ranked with regard to the representativeness of each topic in the hierarchy, based on two factors: *popularity* and *discriminativeness*. A phrase is *popular* for a topic if it appears frequently in documents containing that topic (e.g., ‘information retrieval’ has better popularity than ‘cross-language information retrieval’ in the Information Retrieval topic). A phrase is *discriminative* of a topic if it is frequent only in the documents about that topic but not in those about other topics (e.g., ‘query processing’ is more discriminative than ‘query’ in the Databases topic).

We use the topic word distributions inferred from our model to estimate the ‘topical’ count $c_{i,P}(t)$ of each phrase P in each document d_i , in a similar way as we estimate the topical count of words in Equation (21):

$$c_{i,P}(t) = c_{i,P}(\pi_t) p(t|P, \pi_t) = c_{i,P}(\pi_t) \frac{\alpha_{\pi_t, \chi_t} \prod_{x \in P} \phi_{t,x}}{\sum_{z=1}^{C_{\pi_t}} \alpha_{\pi_t, z} \prod_{x \in P} \phi_{\pi_t/z, x}} \quad (23)$$

Let the conditional probability $p(P|t)$ be the probability of “randomly choose a document and a phrase that is about topic t , the phrase is P .” It can be estimated as $p(P|t) = \frac{1}{D} \sum_{i=1}^D \frac{c_{i,P}(t)}{\sum_{P'} c_{i,P'}(t)}$. The popularity of a phrase in a topic t can be quantified by $p(P|t)$. The discriminativeness can be measured by the log ratio of the probability $p(P|t)$ conditioned on topic t , and the probability $p(P|\pi_t)$ conditioned on its parent topic π_t : $\log \frac{p(P|t)}{p(P|\pi_t)}$.

As studied in Wang *et al.* [28], a good way to combine these two factors is to use their product:

$$r_t(P) = p(P|t) \log \frac{p(P|t)}{p(P|\pi_t)} \quad (24)$$

which has an information-theoretic meaning: the pointwise KL-divergence between two probabilities. Finally, we use $r_t(P)$ to rank phrases in topic t in the descending order.

6. EXPERIMENTS

In this section we first introduce the datasets and the methods used for comparison, and then describe our evaluation on *scalability*, *robustness*, and *interpretability*.

Datasets. Our performance study is on four datasets:

- **DBLP title:** A set of titles of recently published papers in DBLP¹. The set has 1.9M titles, 152K unique words, and 11M tokens.
- **CS abstract:** A dataset of computer science paper abstracts from Arnetminer². The set has 529K papers, 186K unique words, and 39M tokens.
- **TREC AP news:** A TREC news dataset (1998). It contains 106K full articles, 170K unique words, and 19M tokens.
- **Pubmed abstract:** A dataset of life sciences and biomedical topic. We crawled 1.5M abstracts³ from Jan. 2012 to Sep. 2013. The dataset has 98K unique words after stemming and 169M tokens.

We remove English stopwords from all the documents. Documents shorter than 3 tokens are not used for computing the moments because we rely on up to third order moments.

Methods for Comparison. Our method is compared with the following topical hierarchy construction methods.

- **hPAM** – parametric hierarchical topic model. The hierarchical Pachinko Allocation Model [21] is a state-of-the-art parametric hierarchical topic modeling approach. hPAM outputs a specified number of supertopics and subtopics, as well as the associations between them.
- **nCRP** – nonparametric hierarchical topic model. Although more recently published nonparametric models have more capability in document modeling, their scalability is worse than nested Chinese Restaurant Process [11]. So we choose nCRP to represent this category. It outputs a tree with a specified height, but the number of topics is determined by the algorithm. A hyperparameter can be tuned to generate more or fewer topics. In our experiment we tune it to generate an approximately identical number of topics as other methods.
- **splitLDA** – recursively applying LDA. We implement a recursive method described by Pujara and Skomoroch [24]. They use LDA to infer topics for each level, and split the corpus according to the inferred results to produce a smaller corpus for inference with the next level. This heuristic method has the best known scalability so far. For fair comparison of the fundamental computational complexity of different algorithms, we do not use any parallel implementation for all the methods. So we implement splitLDA on top of a fast single-machine LDA inference algorithm [31].
- **CATHY** – recursively clustering word co-occurrence networks. CATHY [28] uses a word co-occurrence network to compress the documents and performs topic discovery through an EM algorithm.
- **STROD** – and its variations RTOD, RTOD₂, RTOD₃. This is our scalable tensor recursive orthogonal decomposition method. We implement several versions to analyze our scalability improvement techniques: (i) RTOD:

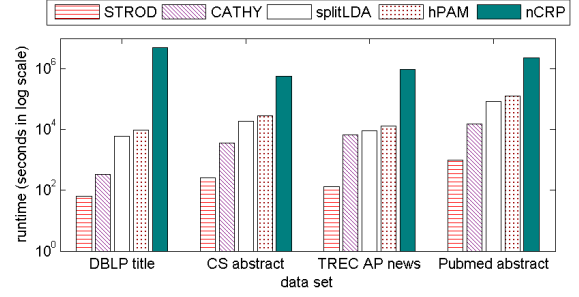


Figure 4: Total runtime on each dataset, $H = 2, C_t = 5$

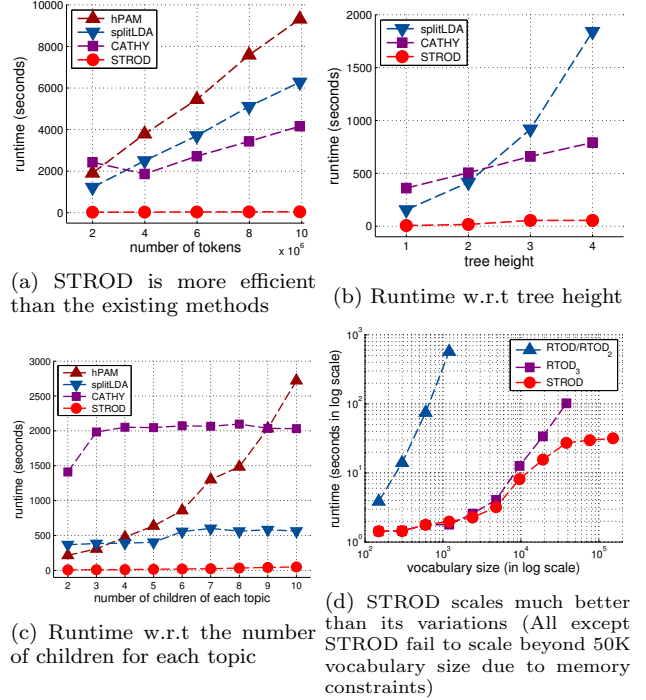


Figure 5: Runtime varying with scale

recursive tensor orthogonal decomposition without scalability improvement (Algorithm 2); (ii) RTOD₂: RTOD plus the technique of avoiding creation of M_2 ; (iii) RTOD₃: RTOD plus the technique of avoiding creation of M_3 ; and (iv) STROD: Algorithm 3 with the full scale-up technique.

We use an optimized Java implementation MALLET [20] for the first three Gibbs sampling-based methods, and set the number of iterations to be 1000, which is the common practice. We implement CATHY and STROD in MATLAB because Java does not have good support with matrix computation and spectral algorithms.

6.1 Scalability

Our first evaluation assesses the scalability of different algorithms, which is one focal point of this paper.

Figure 4 shows the overall runtime in these datasets. STROD is several orders of magnitude faster than the existing methods. On the largest dataset it reduces the runtime from one

¹<http://www.dblp.org>

²<http://www.arnetminer.org>

³<http://www.ncbi.nlm.nih.gov/pubmed>

or more days to 18 minutes. CATHY is the second best method in short documents such as titles and abstracts because it compresses the documents into word co-occurrence networks. But it is still more than 100 times slower than STROD due to many rounds of EM iterations. splitLDA and hPAM rely on Gibbs sampling, and the former is faster because it recursively performs LDA, and considers fewer dependencies in sampling. nCRP is two orders of magnitude slower due to its nonparametric nature.

We then conduct analytical study of the runtime growth with respect to different factors. Figures 5a-5c show the runtime varying with the number of tokens, the tree height and the tree width. We can see that the runtime of STROD grows slowly, and it has the best performance in all occasions. In Figure 5b, we exclude hPAM because it is designed for $H = 2$. In Figure 5c, we use the same number of child topics C_t for each node for all the methods. We exclude nCRP from all these experiments because it takes too long time to finish (>90 hours with 600K tokens).

Figure 5d shows the performance in comparison with the slower variations of STROD. Both RTOD and RTOD₂ fail to finish when the vocabulary size grows beyond 1K, because the third-order moment tensor M_3 requires $O(V^3)$ space to create. RTOD₃ also has limited scalability because the second order moment tensor $M_2 \in \mathbb{R}^{V \times V}$ is dense. STROD scales up easily by avoiding explicit creation of these tensors.

6.2 Robustness

Our second evaluation assesses the robustness of different algorithms. For each dataset, we sample 10,000 documents and run each algorithm 10 times and measure the *variance* among the 10 runs for the same method as follows. Each pair of algorithm runs generate the same number of topics, but their correspondence is unknown. For example, the topic $o/1$ in the first run may be close to $o/3$ in the second run. We measure the KL divergence between all pairs of topics between the two runs, build a bipartite graph using the negative KL divergence as the edge weight, and then use a maximum matching algorithm to determine the best correspondence (top-down recursively). Then we average the KL divergence between matched pairs as the difference between the two algorithm runs. Finally, we average the difference between all $10 \times 9 = 90$ ordered pairs of algorithm runs as the final variance. We exclude nCRP in this section, since even the number of topics is not a constant after each run. Due to space limitation, we report the variance on the first three datasets.

Table 2 summarizes the results: STROD has lowest variance in all the three datasets. The other three methods based on Gibbs sampling have variance larger than 1 in all datasets, which implies that the topics generated across multiple algorithm runs are considerably different.

We also evaluate the variance of STROD when we vary the number of outer and inner iterations N and n . As shown in Figure 6, the variance of STROD quickly diminishes when the number of outer and inner iterations grow to 10. The same trend is true for other datasets. This validates the theoretical analysis of their fast convergence and the guarantee of robustness.

In conclusion, STROD achieves robust performance with small runtime. It is stable and reliable to be used as a hierarchy construction method for large text collections.

Table 2: The variance of multiple algorithm runs in each dataset

Method	DBLP title	CS abstract	TREC AP news
hPAM	5.578	5.715	5.890
splitLDA	3.393	1.600	1.578
CATHY	17.34	1.956	1.418
STROD	0.6114	0.0001384	0.004522

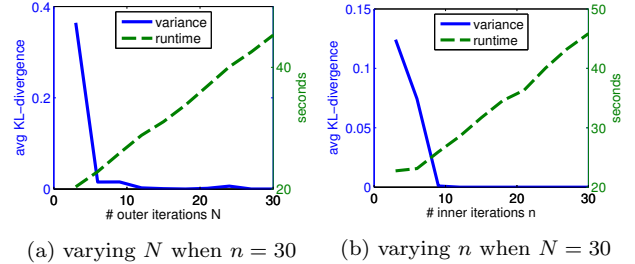


Figure 6: The variance and runtime of STROD when varying # outer and inner iterations N and n (CS abstract)

6.3 Interpretability

Our final evaluation assesses the interpretability of the constructed topical hierarchy, via human judgment. We evaluate hierarchies constructed from DBLP titles and TREC AP news. For simplicity, we set the number of subtopics to be 5 for all topics. For hPAM, we post-process them to obtain the 5 strongest subtopics for each topic. For all the methods we use the same phrase mining and ranking procedure to enhance the interpretability. We do not include nCRP in this study because hPAM has been shown to have superior performance of it [21].

In order to evaluate the topic coherence and parent-child relationship, we use two *intrusion detection* tasks which were proposed in [28] (adopting the idea in [7]):

- **Phrase Intrusion (PI):** X phrases are shown to an evaluator. One is a top-10 phrase from a sibling topic, and the remaining ones come from the top-10 phrases of the same topic. Evaluators are asked to select the intruder phrase, or to indicate that they are unable to make a choice.
- **Topic Intrusion (TI):** Evaluators are shown a parent topic t and X candidate child topics. $X - 1$ of the child topics are actual children of t in the generated hierarchy, and the remaining child topic is not. Each topic is represented by its top-5 ranked phrases. Evaluators are asked to select the intruder child topic, or to indicate that they are unable to make a choice.

For this study we set $X = 4$. 160 Topic Intrusion questions and 200 Phrase Intrusion questions are randomly generated from the hierarchies constructed by these methods. We then calculate the agreement of the human choices with the actual hierarchical structure constructed by the various methods. We consider a higher match between a given hierarchy and human judgment to imply a higher quality hierarchy. For each method, we report the F1 measure of the questions answered ‘correctly’ (matching the method) and consistently by three human judges with CS background.

Figure 7 summarizes the results. STROD is among the best performing methods in both tasks. This suggests that the quality of the hierarchy is not compromised by the strong

scalability and robustness of STROD. As expected, splitLDA and STROD perform similarly in PI task, since they share the same LDA process for one level. However, STROD has a more principled model and theoretically guaranteed inference method to construct the hierarchy. That lead to more meaningful parent-child relations, and thus better performance in TI task.

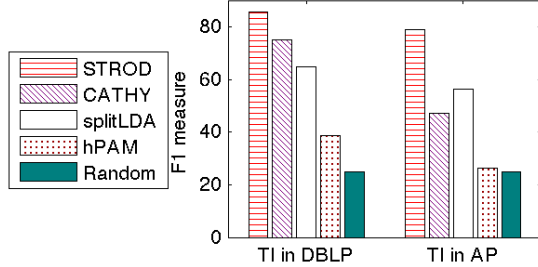


Figure 7: Phrase intrusion and topic intrusion study

A subset of the hierarchy constructed from CS abstract is presented in Figure 8. For each non-root node, we show the top ranked phrases. Node o/1 is about ‘data’, while its children involves database, data mining and bioinformatics. The lower the level is, the more pure the topic is, and the more multigrams emerge ahead of unigrams in general.

7. CONCLUSIONS

In this work, we tackle the scalability and robustness challenge of topical hierarchy construction from large-scale text data. We design a novel framework to build the hierarchy recursively. A nice property of our hierarchical topic model permits dividing the inference problem into smaller subproblems. For robust inference, we leverage a theoretically promising tensor orthogonal decomposition technique. Utilizing the special structure of the tensor in our task, we dramatically overhaul the standard computing procedure to scale up the algorithm. By evaluating our approach on a variety of datasets, we demonstrate a huge computational advantage. Our algorithm generates stable and high-quality topic hierarchy 100-1000 times faster than the state-of-the-art, and the margin grows when the corpus size increases.

Our invention opens up numerous possibilities for future work. On the application side, new systems can be built to support explorative data analysis in multiple granularity, domain knowledge learning, and OLAP in a large scale. On the methodology side, the advantage of STROD can be further fulfilled by parallelization and adaptation to dynamic text collections. We would also like to study how to apply or extend the powerful STROD technique to solve other problems in big data analysis, such as mining latent entity structures in heterogeneous information networks.

8. REFERENCES

- [1] A. Ahmed, Q. Ho, C. H. Teo, J. Eisenstein, E. P. Xing, and A. J. Smola. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *AISTATS*, 2011.
- [2] A. Ahmed, L. Hong, and A. Smola. Nested chinese restaurant franchise process: Applications to user tracking and document modeling. In *ICML*, 2013.
- [3] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y.-K. Liu. A spectral algorithm for latent dirichlet allocation. *arXiv preprint arXiv:1204.6703*, 2012.
- [4] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*, 2012.
- [5] D. M. Blei and J. D. Lafferty. Visualizing Topics with Multi-Word Expressions. *arXiv preprint arXiv:0907.1013*, 2009.
- [6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [7] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. M. Blei. Reading tea leaves: How humans interpret topic models. In *NIPS*, 2009.
- [8] M. Danilevsky, C. Wang, N. Desai, J. Guo, and J. Han. Automatic construction and ranking of topical keyphrases on collections of short documents. In *SDM*, 2014.
- [9] J. Foulds, L. Boyles, C. DuBois, P. Smyth, and M. Welling. Stochastic collapsed variational bayesian inference for latent dirichlet allocation. In *KDD*, 2013.
- [10] B. C. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of SIAM international conference on data mining*, pages 59–70, 2003.
- [11] T. Griffiths, M. Jordan, J. Tenenbaum, and D. M. Blei. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, 2004.
- [12] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.
- [13] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.
- [14] M. Hoffman, D. M. Blei, and D. M. Mimno. Sparse stochastic inference for latent dirichlet allocation. In *ICML*, 2012.
- [15] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, Jan. 2001.
- [16] J. H. Kim, D. Kim, S. Kim, and A. Oh. Modeling topic hierarchies with the recursive chinese restaurant process. In *CIKM*, 2012.
- [17] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, 2006.
- [18] R. V. Lindsey, W. P. Headden, III, and M. J. Stipicevic. A phrase-discovering topic model using hierarchical pitman-yor processes. In *EMNLP-CoNLL*, 2012.
- [19] X. Liu, Y. Song, S. Liu, and H. Wang. Automatic taxonomy construction from keywords. In *KDD*, 2012.
- [20] A. K. McCallum. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- [21] D. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In *ICML*, 2007.
- [22] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.
- [23] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD*, 2008.
- [24] J. Pujara and P. Skomoroch. Large-scale hierarchical topic models. In *NIPS Workshop on Big Learning*, 2012.
- [25] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.
- [26] P. T. P. Tang and E. Polizzi. Feast as subspace iteration accelerated by approximate spectral projection. *arXiv preprint arXiv:1302.0432*, 2013.
- [27] H. M. Wallach. Topic modeling: beyond bag-of-words. In *ICML*, 2006.
- [28] C. Wang, M. Danilevsky, N. Desai, Y. Zhang, P. Nguyen, T. Taula, and J. Han. A phrase mining framework for recursive construction of a topical hierarchy. In *KDD*, 2013.
- [29] C. Wang, X. Yu, Y. Li, C. Zhai, and J. Han. Content coverage maximization on word networks for hierarchical topic summarization. In *CIKM*, 2013.
- [30] X. Wang, A. McCallum, and X. Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM*, 2007.
- [31] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, 2009.
- [32] K. Zhai, J. Boyd-Graber, N. Asadi, and M. L. Alkhouja. Mr. lda: A flexible large scale topic modeling package using

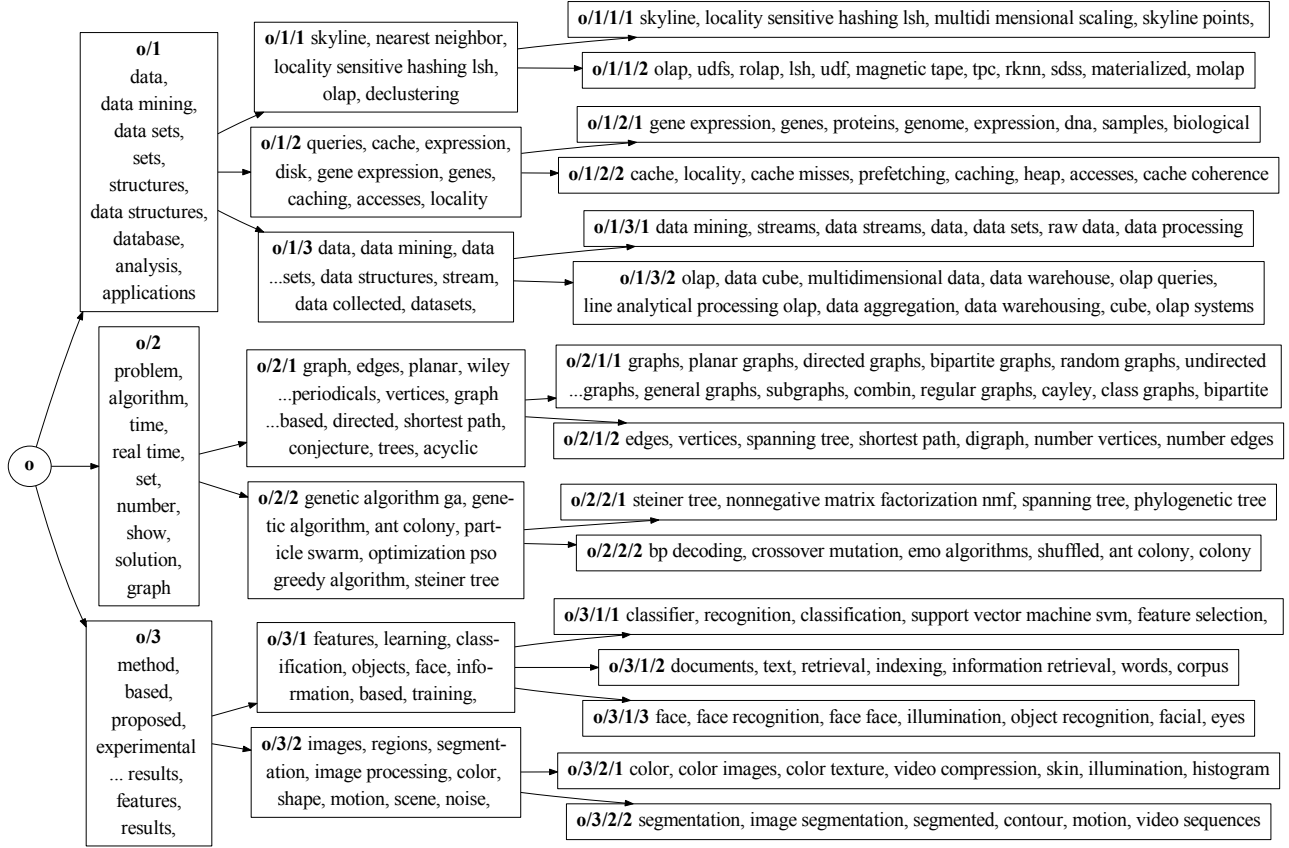


Figure 8: Sample of hierarchy generated by STROD (two phrases only differing in plural/single forms are shown only once)

variational inference in mapreduce. In *WWW*, 2012.

Appendix A. Discussion of Theorem 1

Theorem 1 relies on several non-trivial claims: i) the orthogonal decomposition of \tilde{T} is unique; ii) the power iteration converges robustly and quickly to the eigenpair; and iii) a pair of $(\tilde{\lambda}_z, \tilde{v}_z)$ uniquely determines a pair of (λ_z, v_z) . The first two are proved in Anandkumar *et al.* [4], and the third can be proved by a similar proof of Theorem 4.1 in Anandkumar *et al.* [3]. We omit the details here. To see why these claims are non-trivial, we notice that the decomposition of $M_2 = \sum_{z=1}^k \lambda_z v_z \otimes v_z$ is not unique. If (σ_z, μ_z) are orthonormal eigenpairs of M_2 , then for any orthonormal matrix $O \in \mathbb{R}^{k \times k}$, $M_2 = \sum_{z=1}^k \sigma_z (O\mu_z) \otimes (O\mu_z)$. So there are infinite number of ways of decomposition if we only consider second order moments. This explains why CATHY’s word co-occurrence network model has no robust inference method, since the word co-occurrence information is equivalent to the second order moments.

Anandkumar *et al.* [4] also provides perturbation analysis about Algorithm 1. When N and n are sufficiently large, the decomposition error is bounded by the error ϵ of empirical moments from theoretical moments. The number of required inner loop iterations n grows in a logarithm rate with k , and the outer loop N in a polynomial rate. They also proposed possible stopping criterion to reduce the number of trials of the random restart. Since the number of components k is bounded by a small constant $K \approx 10$ in our task, the power iteration update is very efficient, and we observe that $N = n = 30$ are sufficient.

Appendix B. Hyperparameter Learning

1. Selection of the number of topics. We discuss how to select C_t when the tree width K is given. We first compute the largest K eigenvalues of E_2 in Line 3.2, and then select the smallest k such that the first k eigenpairs form a subspace that is good approximation of E_2 ’s column space. This is similar to the idea of using Principle Component Analysis (PCA) to select a small subset of the eigenvectors as

basis vectors. The *cumulative energy* $g(k)$ for the first k eigenvectors is defined to be $g(k) = \sum_{z=1}^k \sigma_z$. And we choose the smallest value of k such that $\frac{g(k)}{g(K)} > \eta$, and let $C_t = k$. $\eta \in [0, 1]$ controls the required energy of the first k eigenvectors, and can be tuned according to the application. When $\eta = 1$ a full K -branch tree will be constructed. When $\eta = 0$ the tree contains a single root node because $C_o = 0$. Typically η between 0.7 and 0.9 results in reasonable children numbers.

2. Learning Dirichlet prior. First, we note that the individual prior $\alpha_{t,z}$ can be learned by the decomposition algorithm, when the summation $\alpha_{t,0}$ of $\alpha_{t,1}$ to α_{t,C_t} is given to perform the inference for topic t . This already largely reduces the number of hyperparameters that are needed to be given. Large $\alpha_{t,0}$ indicates that t ’s subtopics tend to be mixed together in a document, while small $\alpha_{t,0}$ suggests that a document usually talks about only a few of the subtopics. When $\alpha_{t,0}$ approaches 0, one expects a document to have only one subtopic of t . So $\alpha_{t,0}$ can usually be set empirically according to the prior knowledge of the documents, such as 1 to 100.

If one wants to learn $\alpha_{t,0}$ automatically, we propose a heuristic method hereby. Suppose the data are generated by an authentic $\alpha_{t,0}^*$, and the moments are computed using the same $\alpha_{t,0}^*$, then the decomposition result should satisfy $\sum_{z=1}^{C_t} \alpha_{t,z} = \alpha_{t,0}$ exactly. However, if one uses a different $\alpha_{t,0}$ to compute the moments, the moments could deviate from the true value and result in mismatched $\alpha_{t,z}$. The discrepancy between returned $\sum_{z=1}^{C_t} \alpha_{t,z}$ and initial $\alpha_{t,0}$ indicate how much $\alpha_{t,0}$ deviates from the authentic value. So we can use the following fixed-point method to learn $\alpha_{t,0}$, where δ is learning rate.

1. Initialize $\alpha_{t,0} = 1$;
2. While (not converged)
 - (a) Perform tensor decomposition for topic t to update $\alpha_{t,z}, z = 1, \dots, C_t$;
 - (b) $\alpha'_{t,0} = \sum_{z=1}^{C_t} \alpha_{t,z}$;
 - (c) Update $\alpha_{t,0} \leftarrow \alpha_{t,0} + \delta(\alpha'_{t,0} - \alpha_{t,0})$;